Recurrence - Examples

Syed Tanweer Shah Bukhari

Outline

- Solving Recurrences
  - Iteration Method
  - Substitution Method
  - The Master Theorem

- $T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + 1 & for \quad n > 0 \end{cases}$
- $T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + n & for \quad n > 0 \end{cases}$
- $T(n) = \begin{cases} 1 & for \quad n = 1 \\ 2T(\frac{n}{2}) + 1 & for \quad n > 1 \end{cases}$
- $T(n) = \begin{cases} 1 & for \quad n = 1 \\ aT(\frac{n}{b}) + n & for \quad n > 1 \end{cases}$

Solving Recurrences

- Iteration method
- Substitution method
- Master Method

Solving Recurrences

Iterative method

- Expand the recurrence
- Work some algebra to express as a summation
- Evaluate the summation

$$T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + 1 & for \quad n > 0 \end{cases}$$

$T(n) = T(n-1) + 1$

$T(n-1) = T(n-2) + 1$

$T(n) = T(n-2) +1 +1 = T(n-2) +2$

$T(n) = T(n-3) + 3$

....

$T(n) = T(n-k) + k$

$$T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + 1 & for \quad n > 0 \end{cases}$$

So far we have $n \geq k$, we have

T(n) = T(n-k) + k

What if we have $n = k$?

T(n) = T(n-n) + n

T(n) = n

$$T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + n & for \quad n > 0 \end{cases}$$

T(n) = T(n-1) + n

T(n-1) = T((n-1)-1) + (n-1)

T(n) = T(n-2) + (n-1) + n

T(n) = T(n-3) + (n-2) + (n-1) + n

.....

T(n) = T(n-k) + (n-(k-1)+.....+ (n-3) + (n-2) + (n-1) + n

$$T(n) = \begin{cases} 0 & for \quad n = 0 \\ T(n-1) + n & for \quad n > 0 \end{cases}$$

T(n) = T(n-k) + (n-(k-1)+.....+ (n-3) + (n-2) + (n-1) + n

T(n) = $\sum_{i=n-(k+1)}^{n} i$ + T(n-k)

So far for $n \geq k$, we have

T(n) = $\sum_{i=n-(k+1)}^{n} i$ + T(n-k)

What if $n = k$?

T(n) = $\sum_{i=1}^{n} i$ + T(0) = $\sum_{i=1}^{n} i$ + 0 = $\frac{n(n+1)}{2} \approx n^2$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ 2T(\frac{n}{2}) + 1 & for \quad n > 1 \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{\frac{n}{2}}{2}\right) + 1$$

$$T(n) = 2\left(2T\left(\frac{n}{2^2}\right) + 1\right) + 1$$

$$T(n) = 2^2T\left(\frac{n}{2^2}\right) + 2 + 1 = 2^2T\left(\frac{n}{2^2}\right) + 3$$

$$T(n) = 2^3T\left(\frac{n}{2^3}\right) + 4 + 3 = 2^3T\left(\frac{n}{2^3}\right) + 7$$

$$T(n) = 2^4T\left(\frac{n}{2^4}\right) + 15$$

$\dots$

$$T(n) = 2^kT\left(\frac{n}{2^k}\right) + (2^k + 1)$$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ 2T(\frac{n}{2}) + 1 & for \quad n > 1 \end{cases}$$

So far for $n \geq k$, we have

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + (2^k + 1)$$

What if k = logn => 2^k = n ?

$$T(n) = nT\left(\frac{n}{n}\right) + (n + 1)$$

$$T(n) = n + (n + 1)$$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ aT(\frac{n}{b}) + n & for \quad n > 1 \end{cases}$$

$$T(n) = aT\left(\frac{n}{b}\right) + n$$

$$T\left(\frac{n}{b}\right) = aT\left(\frac{\frac{n}{b}}{b}\right) + \frac{n}{b}$$

$$T(n) = a\left(aT\left(\frac{n}{b^2}\right) + \frac{n}{b}\right) + n$$

$$T(n) = a^2T\left(\frac{n}{b^2}\right) + a\frac{n}{b} + n = a^2T\left(\frac{n}{b^2}\right) + n(\frac{a}{b} + 1)$$

$$T(n) = a^3T\left(\frac{n}{b^3}\right) + a^2\frac{n}{b^2} + n(\frac{a}{b} + 1) \ = a^3T\left(\frac{n}{b^3}\right) + n(\frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$\dots$

$$T(n) = a^kT\left(\frac{n}{b^k}\right) + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \dots \dots + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ aT(\frac{n}{b}) + n & for \quad n > 1 \end{cases}$$

$$T(n) = aT\left(\frac{n}{b}\right) + n$$

$$T\left(\frac{n}{b}\right) = aT\left(\frac{\frac{n}{b}}{b}\right) + \frac{n}{b}$$

$$T(n) = a\left(aT\left(\frac{n}{b^2}\right) + \frac{n}{b}\right) + n$$

$$T(n) = a^2T\left(\frac{n}{b^2}\right) + a\frac{n}{b} + n = a^2T\left(\frac{n}{b^2}\right) + n(\frac{a}{b} + 1)$$

$$T(n) = a^3T\left(\frac{n}{b^3}\right) + a^2\frac{n}{b^2} + n(\frac{a}{b} + 1) \ = a^3T\left(\frac{n}{b^3}\right) + n(\frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$\cdots$

$$T(n) = a^kT\left(\frac{n}{b^k}\right) + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots.. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ aT(\frac{n}{b}) + n & for \quad n > 1 \end{cases}$$

So far for $n \geq k$, we have

$$T(n) = a^k T\left(\frac{n}{b^k}\right) + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

What if $k = \ log_b n \Rightarrow b^k = \text{n}$?

$$T(n) = a^k T\left(\frac{n}{n}\right) + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = a^k T(1) + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = a^k \frac{b^k}{b^k} + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = n \frac{a^k}{b^k} + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = \begin{cases} 1 & for \quad n = 1 \\ aT(\frac{n}{b}) + n & for \quad n > 1 \end{cases}$$

$$T(n) = n\frac{a^k}{b^k} + n(\frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

$$T(n) = n(\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

So with $k = \ log_b n$=> b^k = n , we have

$$T(n) = n(\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \frac{a^{k-2}}{b^{k-2}} + \cdots \ldots .. + \frac{a^2}{b^2} + \frac{a}{b} + 1)$$

What if a = b?

$T(n) = n(k + 1)$

$T(n) = n(log_b n + 1)$

$T(n) = nlog_b n + n$

$T(n) = \Theta\ (nlog_b n)$

Solving Recurrences

- The substitution method
- A.k.a. the “making a good guess method”
- Guess the form of the answer then use induction , then use induction to find the constants and show that solution works

Examples:

- $T(n) = 2T(n/2) + \Theta(n)$ -> $T(n) = \Theta(n \log n)$
- $T(n) = 2T(\lfloor n/2 \rfloor) + n$ -> ???

Example

T(n) = 2T(n/2) +n …………….A

Suppose
T(n) <= n
T(n/2) <= n/2 substitute in equation A
T(n)= 2 (n/2) +n
= n + n which is no equal to T(n) <=n

Suppose
T(n) <= nlgn
T(n/2) <= n/2lgn/2 substitute in equation A
T(n) = 2 n/2 log(n/2) +n
= n lgn- n lg2 +n
=nlgn-n +n
=nlgn

Example (Binary Search)

$T(n) = T(n/2) + 1$ ……………...A

Suppose $T(n) <= lgn$

$T(n/2)<=lg(n/2)$ Substitute in A

$T(n) = lg(n/2) + 1$

$= lgn-lg2 + 1$

$= lgn-1 + 1$

$= lgn$ Which is equal to $T(n)<=lgn$

Suppose $T(n) < n =n$

$T(n/2)<=n/2$

$T(n)= n/2 + 1$

$T(n) =(n+2)/2$ which is not equal to $T(n) <=n$

Example

T(n) = 2 T[(n/2) + 17] + n

we neglect 17, because when *n* is large the difference between *T(n/2)* and *T[(n/2)+17)* is not that large: both cut *n* nearly even in half.

T(n) = 2 T(n/2) + n

Suppose T(n)<=nlgn

T(n/2)<=n/2lgn/2

T(n)= 2n/2lgn/2 +n

= nlgn-nlg2+n

=nlgn-n+n

= lgn

Divide and Conquer Approach

- Many useful algorithms are recursive in structure: to solve a given problem they call themselves recursively one or more times to deal with closely related sub-problems.

- These algorithms typically follow a divide-and-conquer approach: they break the problem into several sub problems that are similar to the original problem but smaller in size, solve the problem recursively, and then combine these solutions to create a solution to the original problem.

Divide and Conquer Approach

- The divide-and-conquer paradigm involves three steps at each level of the recursion:
  - Divide the problem into a number of sub-problems.
  - Conquer the sub-problems by solving them recursively. If the sub-problems sizes are small enough.
  - Combine the solution to the sub problems for the original problem.

Divide and Conquer Approach

- Suppose we divide the problem into a sub-problems, each of which is 1/b the size of the original.
- If we take D(n) time to divide the problem into sub-problems and C(n) time to combine the solutions to the sub-problems into the solution of the original problem, we get the recurrence

$$T(n) = \begin{cases} O(n) & for \quad n \leq c \\ aT(\frac{n}{b}) + D(n) + C(n) & otherwise \end{cases}$$

The Master Theorem

- Given: a *divide and conquer* algorithm
  - An algorithm that divides the problem of size n into a sub-problems, each of size n/b
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved sub-problems) be described by the function $f(n)$
- Then, the Master Theorem gives us a cookbook for the algorithm's running time:

The Master Theorem

- If $T(n) = aT(\frac{n}{b}) + f(n)$ then

- $$T(n) = \begin{cases} \Theta\left(n^{log_b a}\right) & f(n) = O(n^{log_b a - \varepsilon}) \\ \Theta\left(n^{log_b a} log n\right) & f(n) = \Theta(n^{log_b a}) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{log_b a + \varepsilon}\right) AND \\ & af\left(\frac{n}{b}\right) < cf(n) for\ large\ n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

Example (Quick Sort)

T(n)=2T(n/2) + n

a=2 b=2 f(n)=n

$n^{log_b a} = n^{log_2 2} = n$ which is equal to f(n), Condition of Case-II are satisfied

$$T(n) = \Theta(n^{log_b a} logn)$$
$$T(n) = \Theta(nlogn)$$

Equal Case II
> Case I
< Case III

Example (Binary Search)

T(n)=2T(n/2) + 1

a=2 b=2 f(n)=1

$n^{log_b a} = n^{log_2 1} = n^0 = 1$ which is equal to f(n), Condition of Case-II are satisfied

$$T(n) = \Theta(n^{log_b a} logn)$$
$$T(n) = \Theta\left(n^{log_2 1} logn\right)$$
$$T(n) = \Theta(n^0 logn)$$

Equal Case II
> Case I
< Case III

Example

T(n) = 9T(n/3) + n

a=9 b=3 f(n) = n

$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

Since f(n) = $\Theta(n^{\log_3 9-\varepsilon})$, where $\varepsilon = 1$, Case-I applies:

$T(n) = \Theta\left(n^{\log_b a}\right)$ when $f(n) = \Theta(n^{\log_b a-\varepsilon})$

Thus the solution is T(n) = $\Theta(n^2)$

Example

T(n) = 3T(n/4) + nlgn

a=3 b=4 f(n) = nlgn

$n^{log_b a} = n^{log_4 3} = \Theta(n^{0.793})$ where ε approx. = 0.2

Since f(n) = $\Theta(n^{log_4 3+\varepsilon})$, where $\varepsilon = 0.2$, Case-III applies:

Case III if $f(n) = \Omega\left(n^{log_b a+\varepsilon}\right)$ for constant $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some constant $c<1$ and all sufficient large n, then

$$T(n) = \Theta\ (f(n))$$

a f(n/b)=3(n/4)lg(n/4) ≤3/4 nlgn=c f(n) for c=3/4

af(n/b)=3/4nlg(n/4)

Thus the solution is T(n) =$\Theta(nlogn)$

f(n)= nlgn

af(n/b)=a n/b log (n/b)

3f(n/4) = 3(n/4)lgn/4

Example

$\mathrm{T(n)} = 4\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \mathrm{n}^3$

a=4 b=2 $\mathrm{f(n)} = \mathrm{n}^3$

$n^{\log_b a} = n^{\log_2 4} = \Theta(n^2)$ where $\varepsilon = 1$

Since f(n) = $\Theta(n^{\log_2 4+\varepsilon})$, where $\varepsilon = 1$, Case-III applies:

Case III if $f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right)$ for constant ε >0, and if

af(n/b) ≤ *cf(n)* for some constant *c<1* and all sufficient large *n*, then

$$\mathrm{T(n)} = \Theta\ (\mathrm{f(n)})$$

a f(n/b)=4 $(n/2)^3 \le \frac{4}{2}n^3$=c f(n) for c=4/2

af(n/b)=$\frac{4}{8}n^3 \le \frac{4}{2}n^3$ =c f(n)

af(n/b)=$\frac{1}{2}n^3 \le 2\,n^3$ =c f(n)

Thus the solution is T(n) =$\Theta(n^3)$

f(n)= n^3
4f(n/2)=4 $(n/2)^3$
af(n/b) = $a(n/b)^3$

Example

$\mathrm{T(n)} = \mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \mathrm{n}^2$

a=1 b=2 $\mathrm{f(n)} = \mathrm{n}^2$

$n^{log_b a} = n^{log_2 1} = n^0$ where $\varepsilon = 2$

Since f(n) = $\Theta(n^{log_2 1+\varepsilon})$, where $\varepsilon = 2$, Case-III applies:

Case III if $f(n) = \Omega\left(n^{log_b a+\varepsilon}\right)$ for constant ε >0, and if *af(n/b)* ≤ *cf(n)* for some constant *c<1* and all sufficient large *n*, then

$$\mathrm{T(n)} = \Theta\ (\mathrm{f(n)})$$

a f(n/b)=$(\frac{\mathrm{n}}{2})^2 \leq \frac{1}{2}n^2$=c f(n)

af(n/b)=$\frac{1}{4}n^2 \leq \frac{1}{2}n^2$ =c f(n) for c=1/2

Thus the solution is T(n) =$\Theta(n^2)$

f(n)= n^2
1f(n/2)= $(n/2)^2$
af(n/b) = $a(n/b)^2$